WEEK 2

Pre-processing Data in Python

Data Pre-processing

Also known as:

Data Cleaning, Data Wrangling

The process of converting or mapping data from the initial "raw" form into another format, in order to prepare the data for further analysis.

Learning Objectives

- Identify and handle missing values
- Data Formatting
- Data Normalization (centering /scaling)
- Data Binning
- Turning Categorical values to numeric variables

Simple Dataframe Operations

```
df["symboling"]        df["body-style"]
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0
5	2	?	audi	gas	std	two	sedan	fwd	front	99.8	...	136	mpfi	3.19	3.40	8.5
6	1	158	audi	gas	std	four	sedan	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.5
7	1	?	audi	gas	std	four	wagon	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.5
8	1	158	audi	gas	turbo	four	sedan	fwd	front	105.8	...	131	mpfi	3.13	3.40	8.3
9	0	?	audi	gas	turbo	two	hatchback	4wd	front	99.5	...	131	mpfi	3.13	3.40	7.0

Simple Dataframe Operations

```
df["symboling"]=df["symboling"]+1
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio
0	4	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0
1	4	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0
2	2	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0
3	3	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0
4	3	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0
5	3	?	audi	gas	std	two	sedan	fwd	front	99.8	...	136	mpfi	3.19	3.40	8.5
6	2	158	audi	gas	std	four	sedan	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.5
7	2	?	audi	gas	std	four	wagon	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.5
8	2	158	audi	gas	turbo	four	sedan	fwd	front	105.8	...	131	mpfi	3.13	3.40	8.3
9	1	?	audi	gas	turbo	two	hatchback	4wd	front	99.5	...	131	mpfi	3.13	3.40	7.0

Dealing with Missing Values in Python

Missing Values

- What is missing value?
- Missing values occur when no data value is stored for a variable (feature) in an observation.
- Could be represented as "?", "N/A", 0 or just a blank cell.

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front

How to deal with missing data?

Check with the data collection source

Drop the missing values

- drop the variable
- drop the data entry

Replace the missing values

- replace it with an average (of similar datapoints)
- replace it by frequency
- replace it based on other functions

Leave it as missing data

Question

how would you deal with missing values for categorical data

- [x] replace the missing value with the mode of the particular column

Correct
correct, the mode is the value that appears most often

- [ ] replace the missing value with the mean of the particular column
- [x] replace the missing value with the value that appears most often of the particular column

Correct
correct, this is called the mode

Skip

Continue

How to drop missing values in Python

- Use `dataframes.dropna()`:

highway-mpg	price
...	...
20	23875
22	NaN
29	16430
...	...

highway-mpg	price
...	...
20	23875
29	16430
...	...

`axis=0` drops the entire row
`axis=1` drops the entire column

```
df.dropna(subset=["price"], axis=0, inplace = True)
```

```
df = df.dropna(subset=["price"], axis=0)
```

Don't Forget

```
df.dropna(subset=["price"], axis=0)
```

```
df.dropna(subset=["price"], axis=0, inplace = True)
```

http://pandas.pydata.org/

Question

what does the following line of code do to the dataframe **df**:

```
df.dropna(axis=0)
```

- ( ) replaces all values **nan** with the mean
- (•) drops all rows that contain a **nan**
- ( ) drops all columns that contain a **nan**

Correct
correct

Skip
Continue

How to replace missing values in Python

Use `dataframe.replace(missing_value, new_value)`:

normalized-losses	make
...	...
164	audi
164	audi
NaN	audi
158	audi
...	...

normalized-losses	make
...	...
164	audi
164	audi
162	audi
158	audi
...	...

```
mean = df["normalized-losses"].mean()
df["normalized-losses"].replace(np.nan, mean)
```

Practice Quiz: Dealing with Missing Values in Python

Practice Quiz • 6 min • 2 total points

Congratulations! You passed!

Grade received 100% **To pass** 66% or higher

Go to next item

1. How would you access the column "symboling” from the dataframe **df?**

1 / 1 point

- (selected)

```
df["symboling"]
```

-

```
df=="symboling"
```

Practice Quiz: Dealing with Missing Values in Python

Practice Quiz • 6 min • 2 total points

○
```
df=="symboling"
```

✓ **Correct**

correct

2. What is the correct symbol for missing data?

1 / 1 point

◉ nan

○ no-data

✓ **Correct**

correct

Data Formatting in Python

Data Formatting

- Data are usually collected from different places and stored in different formats.
- Bringing data into a common standard of expression allows users to make meaningful comparison.

Non-formatted:

- confusing
- hard to aggregate
- hard to compare

City
NY
New York
N.Y
N.Y

City
New York
New York
New York
New York

Formatted:

- more clear
- easy to aggregate
- easy to compare

Applying calculations to an entire column

- Convert ”mpg” to “L/100km” in Car dataset.

city-mpg
21
21
19
...

→

city-L/100km
11.2
11.2
12.4
...

```
df[”city-mpg”]= 235/df[“city-mpg”]


df.rename(columns={”city_mpg”: “city-L/100km”}, inplace=True)
```

Incorrect data types

- Sometimes the wrong data type is assigned to a feature.

```
df["price"].tail(5)

200    16845
201    19045
202    21485
203    22470
204    22625
Name: price, dtype: object
```

Data Types in Python and Pandas

- There are many data types in pandas
- Objects : “A”, “Hello”..
- Int64 : 1,3,5
- Float64 : 2.123, 632.31,0.12

Question

what task does the following line of code perform:

```
df[["price"]] = df[["price"]].astype("float")
```

- ( ) cast the column price to a **float**
- ( ) cast the column price to an **int**
- ( ) cast the column float to a price

Correct

correct

Skip

Continue

Correcting data types

To *identify* data types:

- Use **`dataframe.dtypes()`** to identify data type.

To *convert* data types:

- Use **`dataframe.astype()`** to convert data type.

Example: convert data type to integer in column “price”

```
df["price"] = df["price"].astype("int")
```

Practice Quiz: Data Formatting in Python

Practice Quiz • 3 min • 1 total point

Congratulations! You passed!

Grade received 100% **To pass** 60% or higher

Go to next item

1. How would you cast each element in the column **"price"** to an integer?

1 / 1 point

- [ ]
```
df["price"] = int(df["price"])
```

- [x]
```
df["price"] = df["price"].astype("int")

```

Data Normalization in Python

Data Normalization in Python

Data Normalization

- Uniform the features value with different range.

length	width	height
168.8	64.1	48.8
168.8	64.1	48.8
171.2	65.5	52.4
176.6	66.2	54.3
176.6	66.4	54.3
177.3	66.3	53.1
192.7	71.4	55.7
192.7	71.4	55.7
192.7	71.4	55.9

scale	[150,250]	[50,100]	[50,100]
impact	large	small	small

Data Normalization

age	income
20	100000
30	20000
40	500000

age	income
0.2	0.2
0.3	0.04
0.4	1

Not-normalized

- "age" and "income" are in different range.
- hard to compare
- "income" will influence the result more

Normalized

- similar value range.
- similar intrinsic influence on analytical model.

Methods of normalizing data

Several approaches for normalization:

① $$x_{new} = \frac{x_{old}}{x_{max}}$$

Simple Feature scaling

② $$x_{new} = \frac{x_{old} - x_{min}}{x_{max} - x_{min}}$$

Min-Max

③ $$x_{new} = \frac{x_{old} - \mu}{\sigma}$$

Z-score

Question

z-score values typically range between 0 to 1

- (selected) False
- True

Correct
correct

Skip

Continue

Simple Feature Scaling in Python

With Pandas:

length	width	height
168.8	64.1	48.8
168.8	64.1	48.8
180.0	65.5	52.4
...	...	...

→

length	width	height
0.81	64.1	48.8
0.81	64.1	48.8
0.87	65.5	52.4
...	...	...

```
df["length"] = df["length"]/df["length"].max()
```

Min-max in Python

With Pandas:

length	width	height
168.8	64.1	48.8
168.8	64.1	48.8
180.0	65.5	52.4
...	...	...

→

length	width	height
0.41	64.1	48.8
0.41	64.1	48.8
0.58	65.5	52.4
...	...	...

```
df["length"] = (df["length"]-df["length"].min())/
               (df["length"].max()-df["length"].min())
```

Z-score in Python

With Pandas:

length	width	height
168.8	64.1	48.8
168.8	64.1	48.8
180.0	65.5	52.4
...	...	...

length	width	height
-0.034	64.1	48.8
-0.034	64.1	48.8
0.039	65.5	52.4
...	...	...

```
df["length"] = (df["length"]-df["length"].mean())/df["length"].std()
```

Practice Quiz: Data Normalization in Python
Practice Quiz • 3 min • 1 total point

Congratulations! You passed!

Grade received 100% **To pass** 60% or higher

Go to next item

1. What is the maximum value for feature scaling?

1 / 1 point

1

Correct
We scale the feature to a value between 0 and 1 so the maximum value should be 1.

Binning in Python

Binning

- Binning: Grouping of values into "bins"
- Converts numeric into categorical variables
- Group a set of numerical values into a set of "bins"
- "price" is a feature range from 5,000 to 45,500 (in order to have a **better representation** of price)

price: 5000, 10000,12000,12000, 30000, 31000, 39000, 44000,44500

bins: low Mid High

Binning in Python pandas

price
13495
16500
18920
41315
5151
6295
...

→

price	price-binned
13495	Low
16500	Low
18920	Medium
41315	High
5151	Low
6295	Low
...	...

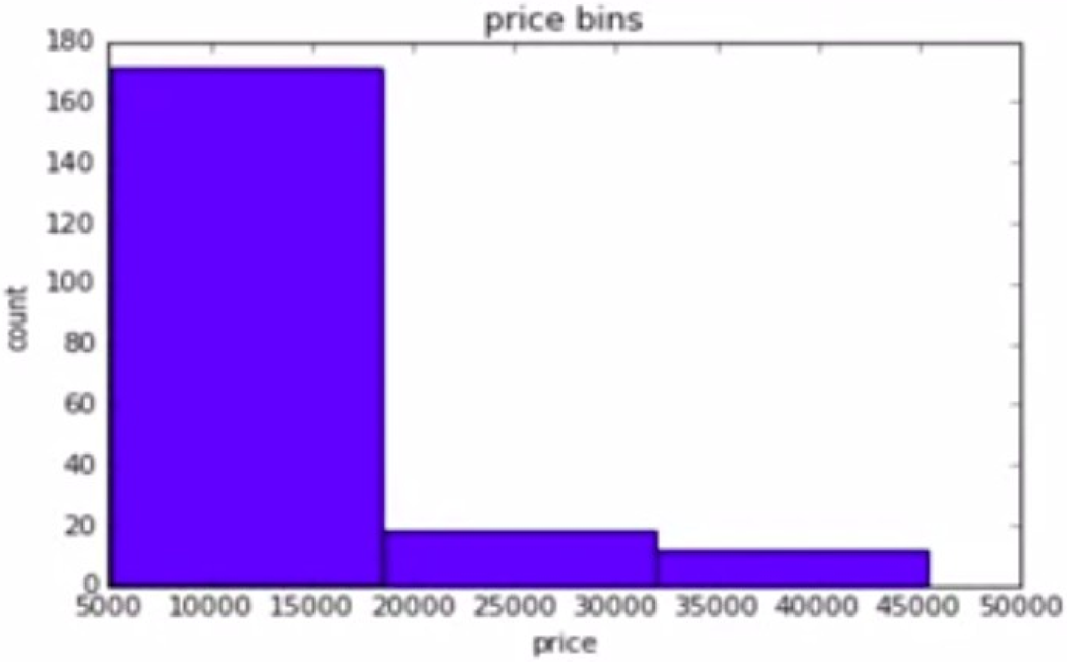
```
bins = np.linspace(min(df["price"]), max(df["price"]), 4)

group_names = ["Low", "Medium", "High"]

df["price-binned"] = pd.cut(df["price"], bins, labels=group_names, include_lowest=True )
```

Visualizing binned data

- E.g., Histograms



Turning categorical variables into quantitative variables in Python

IBM Developer

SKILLS NETWORK

Categorical Variables

Problem:

- Most statistical models cannot take in the objects/strings as input

Car	Fuel	...
A	gas	...
B	diesel	...
C	gas	...
D	gas	...

Categorical → Numeric

Solution:

- Add dummy variables for each unique category
- Assign 0 or 1 in each category

Car	Fuel	...	gas	diesel
A	gas	...	1	0
B	diesel	...	0	1
C	gas	...	1	0
D	gas	...	1	0

"One-hot encoding"

Dummy variables in Python pandas

- Use pandas.get_dummies() method.
- Convert categorical variables to dummy variables (0 or 1)

fuel
gas
diesel
gas
gas

→

gas	diesel
1	0
0	1
1	0
1	0

```
pd.get_dummies(df['fuel'])
```

Practice Quiz: Turning categorical variables into quantitative variables in Python

Practice Quiz • 3 min • 1 total point

Congratulations! You passed!

Grade received 100%   **To pass** 60% or higher

Go to next item

1. Why do we convert values of Categorical Variables into numerical values?

1 / 1 point

- (●) Most statistical models cannot take in objects or strings as inputs
- ( ) To save memory

Correct

correct

Lesson Summary

In this lesson, you have learned how to:

Identify and Handle Missing Values: Drop rows with incomplete information and impute missing data using the mean values.

Understand Data Formatting: Wrangle features in a dataset and make them meaningful for data analysis.

Apply normalization to a data set: By understanding the relevance of using feature scaling on your data and how normalization and standardization have varying effects on your data analysis.

Mark as completed